The School of St Jude

ST. JUDE'S GIRLS' SECONDARY SCHOOL.

SCHEME OF WORK FOR: COMPUTER SCIENCE **CLASS:** FORM TWO **TERM :** TWO **YEAR:** 2026 **TEACHER'S NAME:** ELIZABETH A. MALLYA (Eng.)

MAIN COMPETENCE:	**Demonstrate mastery of the basic principles of Computer Science**
SPECIFIC COMPETENCE	**Demonstrate an understanding of basic principles of computer programming (using an appropriate structured programming language such as C, python, etc.**

MAIN COMPETENCES	SPECIFIC COMPETENCES	LEARNING ACTIVITIES	SPECIFIC LEARNING ACTIVITIES	MONTH	WEEK	PRIODS	TEACHING METHODS & LEARNING METHODS	TEACH/LEARN RESOURCES	TEACH/LEARN MATERIAL	ASSESSMENT CRITERIA	REMARKS
		(c) Install and configure the selected programming language (compiler/ interpreter, text editor, IDE, Debugger)+C18:C31	(i) Students install a text editor or IDE (e.g., VS Code, PyCharm) and configure basic settings (theme, file paths).		28		Demonstration and Practical work: Through demonstration and practical work teacher to guide students to install and configure the selected programming language tools. **Method**: Hands-on practical task + Teacher guidance, Demonstration + Peer-assisted learning	Flash disks with executable files of the selected programming languages.	Executable installer files on flash disk; Setup guide handout	Can master the environment of the selected program; Can install and configure programming tools	
			(ii) Learners install and test the language interpreter or compiler (e.g., Python, GCC) by running a simple command in terminal/command prompt.						Command-line reference card	Can successfully run compiler/interpreter commands	
2.1 Demonstrate an understanding of basic principles of computer programming (using an appropriate structured programming language such as C, python, etc.		(d) Use programming tools of selected programming language to write a program (compile/run and debug a simple program)	**The concept of C programming**: Meaning of programming with C, Setting uo a local development for C programming	**JULY**		*2*	**Practical demonstations**: A teacher to guide students to set up a local development environment for C programming.	Installed IDE for C.	Video tutorial on setting up a local C development environment; IDE installation guide	Can set up a working C development environment	
			Writing and running a simple C program: To write, compile and execute a basic C program.		28		**Practical demonstration**: A teacher to guide students to write a simple C program using Windows Computer with Notepad++ or VS Code.	A code editor	Source code file — simple program printing 'Habari Africa' or 'Hello World'; Terminal output showing successful compilation and execution	Can write, compile and run a basic C program	
		(e) Use variables,	**Basic syntax and structure of a C program:** Structure of a C program, Writing and using comments, Displaying output (printf) and taking inputs (scanf), Understanding escape	**JULY**			**Practical demonstration**: A teacher to guide students to write a simple C program using Windows Computer with Notepad++ or VS Code.	Computers; Code editor; Projector	Handout: C program structure diagram; Code examples with comments	Can correctly structure a C program; Can use printf, scanf and escape sequences	

2.0 Demonstrate mastery of the basic principles of Computer Science		constants, and data types of a selected programming language in a program (operators and expressions)	**Variables, data types and constants:** Declaring and initializing variables in C, Basic Data types(int, float, char, double), Constants and #define Macros, Type conversion and type casting.	JUL	28	*2*	**Practical work:** Through practical work teacher to guide students to use variables and data types of a selected programming language	Computers; Code editor; Projector	Handout: Data types reference table; Code samples	Can declare variables and use correct data types; Can define constants using #define	
			Operators and expressions: Arithmetic operators, Relational operators, Assignment operators, Operator precedence in C, Increment (++) and decrement (--) operators.	JULY	29	*2*	**Practical work:** Through practical work teacher to guide students to use arithmetic operators, relational operators, Assignment operators, Increment (++) and decrement (--) in C programming	Computers; Code editor; Projector	Handout: Operator precedence table; Code examples	Can correctly use arithmetic, relational and assignment operators; Understands operator precedence	
	2.1 Demonstrate an understanding of basic principles of computer programming (using an appropriate structured programming language such as C, python, etc.	(f) Use syntax and constructs of the selected programming language to write programs branching	**Control Flow in C:** Introduction to control flow in C, Selection flow, Loops, Using break and continue in Loops.				**Practical work:** Through practical work teacher to guide students to use control flow through the use of loops, breaks and continue in loops for C programming.	Computers; Code editor; Projector	Flowchart diagrams of control structures; Code examples; Handout	Can write C programs using if/else, switch and loop constructs; Can use break and continue	
			Functions in C programming: Defining function, Calling a function, Function paraments and return types, Void function and return statement.	JULY	29	2	**Practical demonstations:** A teacher to guide students to create and declare a function in C pogramming, with function header and function body(Return type, Function name, parameters, function body)	Computers; Code editor; Projector	Code samples illustrating function definition and calling; Handout on function syntax	Can define, declare and call functions in C; Can use parameters and return values	
			Arrays and Strings: One-dimensional arrays, two-dimensional arrays (matrices), Introduction to strings and string handling.				A teacher to guide students through **practical demonstration** to declare one-dimensioanal array, two-dimensional array, initialize a one and two-dimensional array, declare a string, initialize a string and string handling.	Computers; Code editor; Projector	Code examples for 1D arrays, 2D arrays and strings; Reference handout	Can declare, initialize and use 1D and 2D arrays; Can declare and manipulate strings	
		(g) Debugging computer programs	**File handling**: Introduction to file handling in C, Reading from and writing to file in C, Working with text files.	JULY	30	*2*	**Practical Demonstration:** A teacher to guide students to use basic file handling functions such as fopen(), f(close), fprintf(), and fscanf(), working with text files	Computers; Code editor; Projector	Code examples demonstrating file open/read/write; Sample text files	Can open, read and write to text files using C file handling functions	
2.0 Demonstrate mastery of the basic principles of Computer Science			**Debugging and error handling:** Understanding compilation errors, Debugging techniques in C, Using debugging tools,				A teacher to guide students to understand compilation errors, common errors in C programming.	Computers; Code editor with debugger; Projector	Sample programs with intentional errors; Error message reference list	Can identify and fix syntax errors; Can use debugging tools and print statements to trace errors	
							Guided Practice: Teacher guides students to understand compilation errors and common errors in C programming.	Computers; Code editor with debugger; Projector	Pre-written buggy programs for students to debug	Can locate and correct syntax and logical errors; Can use print statements and debugger to trace code	

			(ii) Learners debug a program containing simple errors (syntax errors, missing brackets, undefined variables) using the debugger or print statements.			*2*	Practical Work: Students practise debugging erroneous programs step by step. Methods: Practical coding exercise + Step-by-step guidance; Problem-solving activity + Teacher-supported debugging.	Computers; Code editor with debugger; Projector	Pre-written buggy programs for students to debug	Can locate and correct syntax and logical errors; Can use print statements and debugger to trace code	

JULY WEEK 31 & 32 - F2 PRE NECTA ONE / REGIONAL EXAMINATIONS

2.0 Demonstrate mastery of the basic principles of Computer Science	2.1 Demonstrate an understanding of basic principles of computer programming (using an appropriate structured programming language such as C, python, etc.	(c) Install and configure the selected programming language (compiler/ interpreter, text editor, IDE, Debugger)+C18:C31	The concept of Python programming: Introduction to Python programming and its applications, Features and advantages of Python language, Installing Python and setting up the	**AUGUST**	33	*2*	Demonstration & Practical Work: Teacher guides students to install Python and set up the programming environment.	Computers; Flash disks with Python installer; Projector	Python installer file; Setup guide handout; Official Python documentation	Can install Python and configure the IDE; Understands advantages of Python	
			Writing and running a simple Python program.				Practical Demonstration: Teacher demonstrates writing and executing a simple Python script.	Computers; Python IDE (VS Code / IDLE); Projector	Sample Python script; Terminal output	Can write and execute a simple Python program	
		(d) Use programming tools of selected programming language to write a program (compile/run and debug a simple program)	Python basics-syntax and structure: Indentation and formatting, Python identifiers: rules, conventions, and best practices, Using comments, Displaying output with print(), Taking user input with input(), Waiting for user input.			*2*	Practical Demonstration + Guided Notes: Teacher demonstrates Python syntax rules, indentation, and basic I/O.	Computers; Python IDE; Projector	Code examples; Handout on Python syntax rules and identifiers	Can correctly structure a Python program; Can use print() and input(); Understands indentation rules	
		(e) Use variables, constants, and data types of a selected programming language in a program (operators and expressions)	Variables, data types and operators: Declaring and assigning variables, Types conversion (changing data types), Arithmetic operators (Math operations in Python), Comparison and logical operators.	**AUGUST**	34	*2*	Practical Work: Teacher guides students to declare variables, use data types and apply arithmetic/comparison/logical operators in Python.	Computers; Python IDE; Projector	Handout: Python data types table; Code examples with operators	Can declare and assign variables; Can perform type conversion; Can use arithmetic and logical operators	
			Conditional statements and control flow in Python: Understanding the conditional statement control flow, if, if-else and elif statements, Using "and", "or", and "not" in conditions.				Practical Work: Teacher guides students to write programs using conditional statements and boolean operators.	Computers; Python IDE; Projector	Flowchart diagrams of conditional logic; Code examples; Handout	Can write if, if-else and elif statements; Can use logical operators in conditions	
			Writing programs with decision making.	**AUGUST**	34	*2*	Practical Coding Exercise: Students write programs that make decisions based on user input.	Computers; Python IDE; Projector	Problem-based exercises requiring decision-making programs	Can write programs that use conditional logic to make decisions	
			Loops and iteration control flow in Python: The for loop - Iterating over sequences.				Practical Work: Teacher guides students to write for and while loops, and use break and continue statements.	Computers; Python IDE; Projector	Code examples for for/while loops; Handout on loop syntax	Can write for and while loops; Can use break and continue statements	
			The while loop - Repeating based on a condition, Using break and continue statements, Simple loop-based programs.				Practical Work: Teacher guides students to write for and while loops, and use break and continue statements.	Computers; Python IDE; Projector	Code examples for for/while loops; Handout on loop syntax	Can write for and while loops; Can use break and continue statements	

AUGUST 35 & SEPT 36 - F2 PRE NECTA TWO EXAMINATIONS

2.0 Demonstrate mastery of the basic principles of Computer Science	2.1 Demonstrate an understanding of basic principles of computer programming (using an appropriate structured programming language such as C, python, etc.	(f) Use syntax and constructs of the selected programming language to write programs branching	Functions in Python: Defining a function, Calling a function, Functions with parameters (inputs), Functions with return values, Functions with multiple parameters.	**SEPTEMBER**	37	*2*	Practical Demonstration: Teacher guides students to define, call and use functions with parameters and return values.	Computers; Python IDE; Projector	Code samples illustrating function definition and calls; Handout on function syntax	Can define and call functions; Can write functions with parameters and return values	
			Python Lists: Creating a list, acsessing list elements, Modifying list elements, adding and removing list elements, Iterating over lists using loops.				Practical Demonstration: Teacher guides students to create and manipulate lists and iterate over them using loops.	Computers; Python IDE; Projector	Code examples for list operations; Handout: Python list methods	Can create and manipulate Python lists; Can iterate over lists with loops	
			Strings and String manipulation in Python: Understanding strings, and indexing, Common string methods, String formatting using f-strings.				Practical Demonstration: Teacher demonstrates string indexing, methods and f-string formatting.	Computers; Python IDE; Projector	Handout: Common Python string methods; Code examples using f-strings	Can manipulate strings using methods; Can use f-strings for formatting	
		(g) Debugging computer programs	File Handling basics in Python: Introduction to file handling in Python, Opening and reading files (open(), read()), Writing to files, Closing files properly (close()).		38	*2*	Practical Demonstration: Teacher guides students to open, read, write and close files in Python.	Computers; Python IDE; Projector; Sample text files	Code examples for file I/O; Sample text files; Handout on file handling functions	Can open, read and write files using Python file handling functions	
			Error Handling in Python: Types of errors in Python, Using try-except for error handling, Importing and using simple modules in Python.				Guided Practical: Teacher guides students to identify error types and write try-except blocks; introduces importing modules.	Computers; Python IDE; Projector	Sample programs with errors; Handout on error types and try-except syntax; Module examples	Can identify error types; Can write try-except blocks; Can import and use basic Python modules	

WEEK 39 & 40 - F2 PRE NECTA THREE EXAMINATIONS

REVISION FOR PREPARATION FOR FTNA	**General Revisions**	Program structure in C and Python	Revision on: Functions in Python and C: Defining a function, Calling a function, Functions with parameters (inputs), Functions with return values, Functions with multiple parameters	**OCTOBER**	41	4	To lead students for revision in Introduction to C programming through practical activities.	Computers; Python IDE; Projector	Sample source codes of Python or C program	Can write a simple program and identify parts of a program in C program and Python program	
		Data types in C and Python	Revision on: Data types in C programming and in Python programming.	**OCTOBER**	42	4	To lead students to revise on Data types in C programming and in Python programming.	Computers; Python IDE; Projector	Sample source codes of Python or C program	Can write program in C and Pyhton that uses different data types and can identify data types in those languages.	
		Revisions in Troubleshooting, Maintenance, Handling, Problem Solving, Internet and Cybersecurity	General revisions on Form One chapters and Form two chapters	**OCTOBER**	43	4	General revisions on Form Ones and Form two topics of Internet, Cybersecurity.	Form Two TIE Book, Past papers, and summary notes.	Summary notes, TIE Books and Revision Exercises from the TIE Book Form One and Form Two.	Can answer questions from the exercises and Revision Exercises from CS TIE book 1 and book 2.	

WEEK 44 TO WEEK 45 - FTNA EXAMS

WEEK 46 - FORM THREE TOPIC EXTENSION

WEEK 47-49 - SASA PROGRAM

WEEK 50 - END OF SASA PROGRAM AND CLOSSING OF SCHOOL